

Lab Documentation for **jenkins**

Lab 1 : Jenkins Installation

We will be installing jenkins on ubuntu machine. Jenkins is opensource software based upon java and can be installed on any operating system.

Step 1

Before moving on to installing the packages on the server, our system package manager must be updated. Use the following command to ensure your system package manager is up to date:

```
$ sudo apt update
```

Step 2

Install Java

Since Jenkins is written in Java, the first step is to install Java. Install the Java 8 OpenJDK package with the following command:

```
$ sudo apt install openjdk-8-jdk
```

The current version of Jenkins doesn't support Java 10 or more yet. If you have multiple java versions installed on your system then make sure java 8 is the default java version.

To check the version of java on your system, use the following command:

```
$ java -version
```

Step 3

Add the Jenkins Debian Repository

Import the GPG (GnuPG - GNU Privacy Guard) keys of the Jenkins repository using the following wget command:

```
$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
```

The output of the above command should OK which means that the key has been successfully imported and packages from this repository will be considered trusted.

Step 4

Now, add the Jenkins repository to the system with the following command:

```
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
```

Step 5

Install Jenkins

Once the Jenkins repository is enabled and keys and sources are added, update the apt package list:

```
$ sudo apt update
```

Now, install the latest version of Jenkins by using the following command:

```
$ sudo apt install jenkins
```

Once the installation is completed, Jenkins service will start automatically.

We can verify it with the help of following command:

```
$ systemctl status Jenkins
```

We should see something like this:

- jenkins.service - LSB: Start Jenkins at boot time

Loaded: loaded (/etc/init.d/jenkins; generated)

Active: active (exited) since Wed 2019-07-06 1308 PDT; 2min 16s ago

Docs: man:systemd-sysv-generator(8)

Tasks: 0 (limit: 2319)

CGroup: /system.slice/jenkins.service

Step 6

Setting Up Jenkins

To set up the new Jenkins installation, open the browser, type the domain or IP address followed by port 8080, `http://your_ip_or_domain:8080`, and screen (unlock Jenkins screen) similar to the following will be displayed:

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Continue

In the terminal, type the following cat command to see the password:

```
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Copy the password from the terminal and paste it into the "Administrator password" field and then click continue. Now, the screen presents the option of installing suggested plugins or selecting specific plugins:

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.85

After the installation of plugins, it's time to create an admin account to login to Jenkins:

Once created, you will be automatically logged into jenkins dashboard

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Dashboard [Jenkins] x +


localhost:8080


Jenkins


search


Jenkins

ENABLE AUTO REFRESH

 [New Job](#)

 [Manage Jenkins](#)

 [People](#)


 [Build History](#)

Build Queue

No builds in the queue.

Build Executor Status

Welcome to Jenkins! Please [create new jobs](#) to get started.

 [add description](#)

Lab 2 : Git Fundamentals

1) install git

```
#apt-get update
```

```
#apt-get install git-core
```

Three stages on repository are

>> working directory : your current directory

>> staged area (index) : files to be committed

>> history : last committed stage

2) Adding the directory and create it the repository

```
#mkdir vishal-repo
```

```
#cd vishal-repo
```

```
#vim file1
```

```
#git init
```

3) Perform a simple administration task to setup username and email.

this is useful to track the commit made by which user

```
git config --global user.name " vishal saini"
```

```
#git config --global user.email "vishal@ap2v.com"
```

```
#git config --list (to display the information)
```

you can use "git config --local" of individual repo

4) to move the file into staging area

```
#git add filename
```

or

```
#git add . ( for all the files inside the directory)
```

to check the status

```
#git status
```

5) To commit the changes

```
# git commit -m " add the new file"
```

save the commit with SHA1 hash of 40 characters

6) To see the information

```
#git log
```

7)To check the status and add another file

```
#vim f2
```

```
#vim f1
```

```
#git status
```

The output shows one file "f2" as untracked and another file "f1" as modified.

8)To check the difference between working directory and staging area

```
#git diff
```

9)Now add the files in the staging area

```
#git add .
```

```
#git diff --staged
```

this shows the difference between the last commit and last changed file

10) Do the commit for both the files

```
#git commit -m " add f2 and edit f1"
```

11) git log to see the status

```
#git log
```

```
#git log -p to show the preview
```

12)To remove the file from working directory and from staged area

```
#git rm f2
```

```
#git status
```

```
#git commit -m "remove commit"
```

```
#git log
```

*note : git commit without -m will open default editor to do the multiline commit's

13)To perform the checkout, let change the f1 file with some boges information

```
#vi f1
```

Don't add the file into staging area, rather run the command to see the difference

```
#git diff
```

you will see boges changes, to rollback to the last staged file

```
#git checkout -- f1
```

14)lets undo staged file

change the file f1

```
#vim f1
```

```
#git diff >> show the difference
```

```
#git add f1
```

Now the working tree and staged file are same but not the commit, to check

```
#get diff --staged
```

to change the file to the last commit

```
#git reset HEAD f1
```

```
#git checkout -- f1
```

15)To restore the deleted file

```
#git log
```

check the last commit where the file is added and run the following command

```
#git checkout $$$$$ -- filename
```

i.e. git checkout 9ddf3 -- f2

```
#ls
```

Better work is to checkout commit in another branch

```
# git checkout -b test-branch 56a4e5c08
```

...do your thing...

```
$ git checkout master
```

```
$ git branch -d test-branch
```


16) You can also use .gitignore file to ignore or untrack certain files from git

```
#vim abc.py
#mkdir a1
#touch a1/log1.log
#touch a1/log2.log
```

```
#git status >> will show all these files as untracked
#vim .gitignore0be6186043db1bb54f962023bd39b484b3dcf510
>> *.pyc
>> a1/
```

add and commit this .gitignore file so that git will safely ignore ur logs and pyc file.

```
#git add .gitignore
#git commit -m "adding .gitignore file"
```

17) Branching is the concept of designing and running multiple code copies for the same setup.

```
#git branch ggn
#git branch noida
#git branch >> to list all the branches.
# git log --all --decorate --oneline --graph
```

18) checkout one branch

```
#git checkout ggn
modify the f1 file and stage and commit it
#vi f1
#git commit -a -m " new commit in ggn branch"
```

19) if you checkout another branch, old data will reflect

```
#git checkout noida
#cat f1
#vim f1
#git commit -a -m " new commit in noida branch"
```

20) check the status

```
#git log --all --decorate --oneline --graph
```

21) It's time to merge both branches into the master branch

Two methods

- > fast forward merge .. when both master and ggn branch are directly connected to each other

- > 3-way merge .. when the merge is moved into another branch

#git checkout master

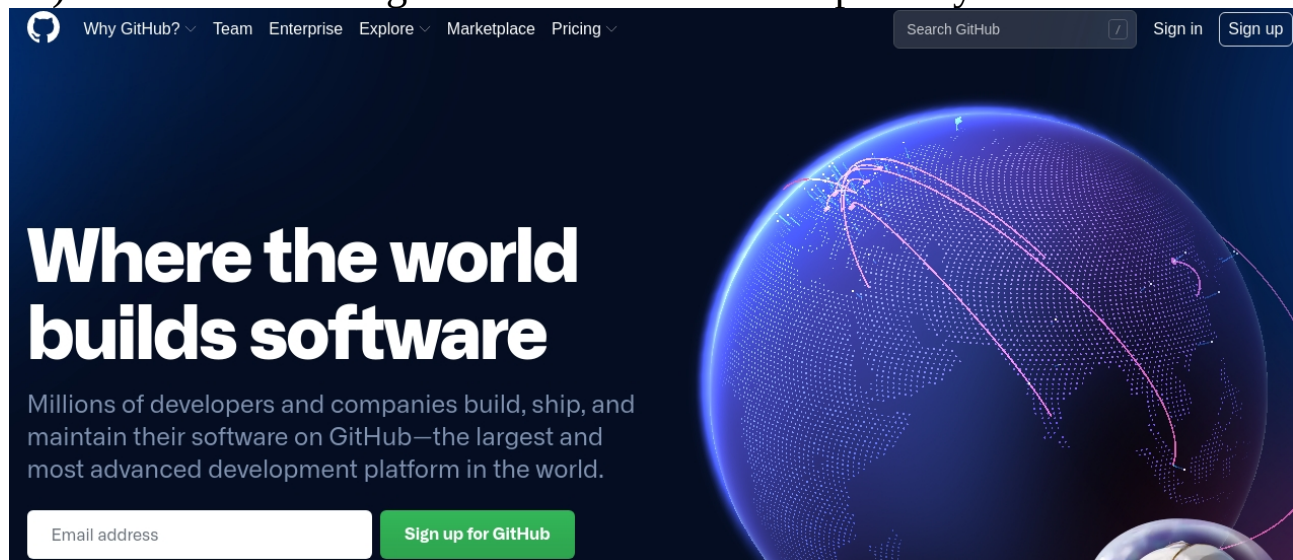
#git diff master..ggn >> to see the difference between two branches

#git merge ggn >> to merge the branch in the master branch

#git branch --merged >> to see the merged branches

#git branch -d ggn >> to delete the branch after merging it into master.

22) Create account in github and create a new repository inside it.



To clone the repository on your local laptop command is

git clone https://github.com/vickybulti/vishalsaini.git

git remote >> to see all our remote, default one is origin

git log --all --decorate --online --graph >> to check the commit history

origin/master is remote tracking branch which tells us what master branch looks at remote origin

origin/master and origin/head and master pointing to same commit

meaning everything is in sync on

both local and remote branches.

1.

23) Git fetch.

Create new commit on github by adding new file and commit it.

2 commit at origin and 1 commit on local system.

to fetch

git fetch origin >> will only fetch the info and will not update any files on localsystem.

to check

git status >>> clear error about two different commit count on local and remote origin

git log --all --decorate --online --graph

Have to merge the origin/master to local master.

git merge origin/master

24) to push data from local repo to remote origin

modify some data in file.

#vi filename

#git add filename

#git commit -m " new data added in file"

#git push origin master
(remotename) (branchname)

Lab 3 : Integrating Github with Jenkins

Ref: <https://www.javatpoint.com/github-setup-for-jenkins>

Step 1

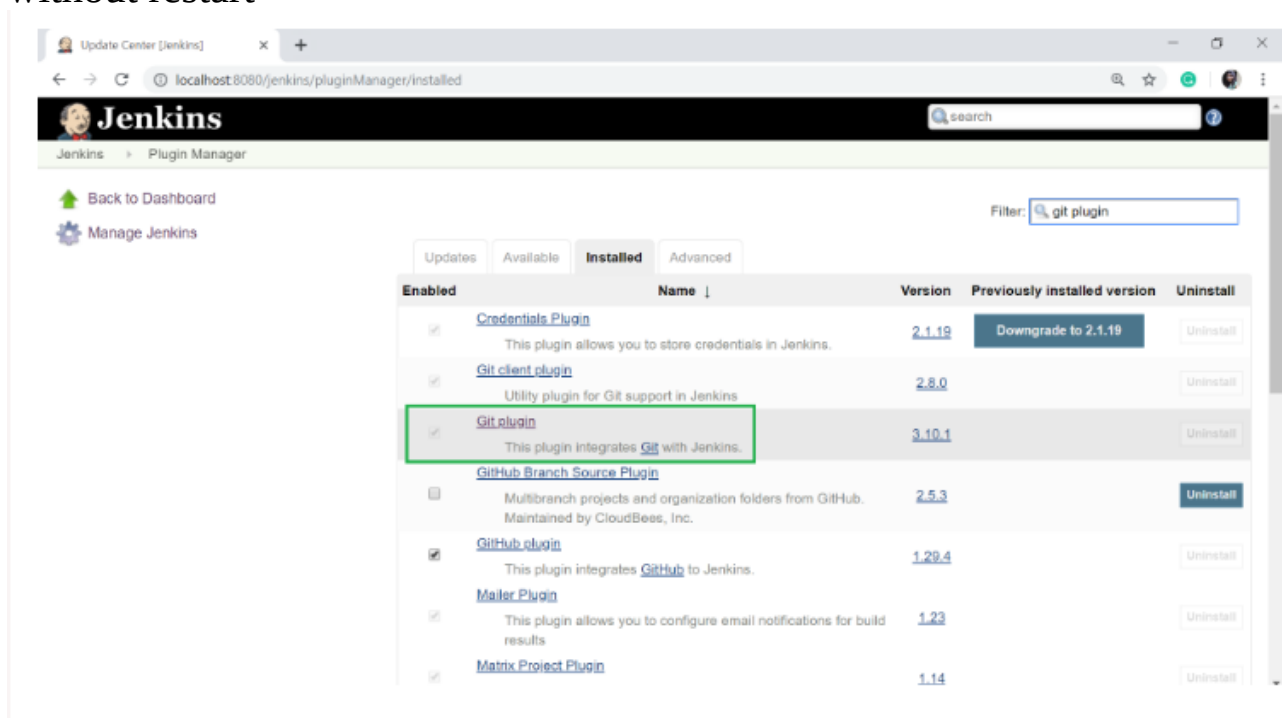
Connecting a GitHub private repository to a private instance of Jenkins can be tricky.

To do the GitHub setup, make sure that internet connectivity is present in the machine where Jenkins is installed.

In the Home screen of the Jenkins (Jenkins Dashboard), click on the Manage Jenkins option on the left hand side of the screen.

click on the Manage Plugins option.

Select Available and search for git plugin, select the same and install them without restart

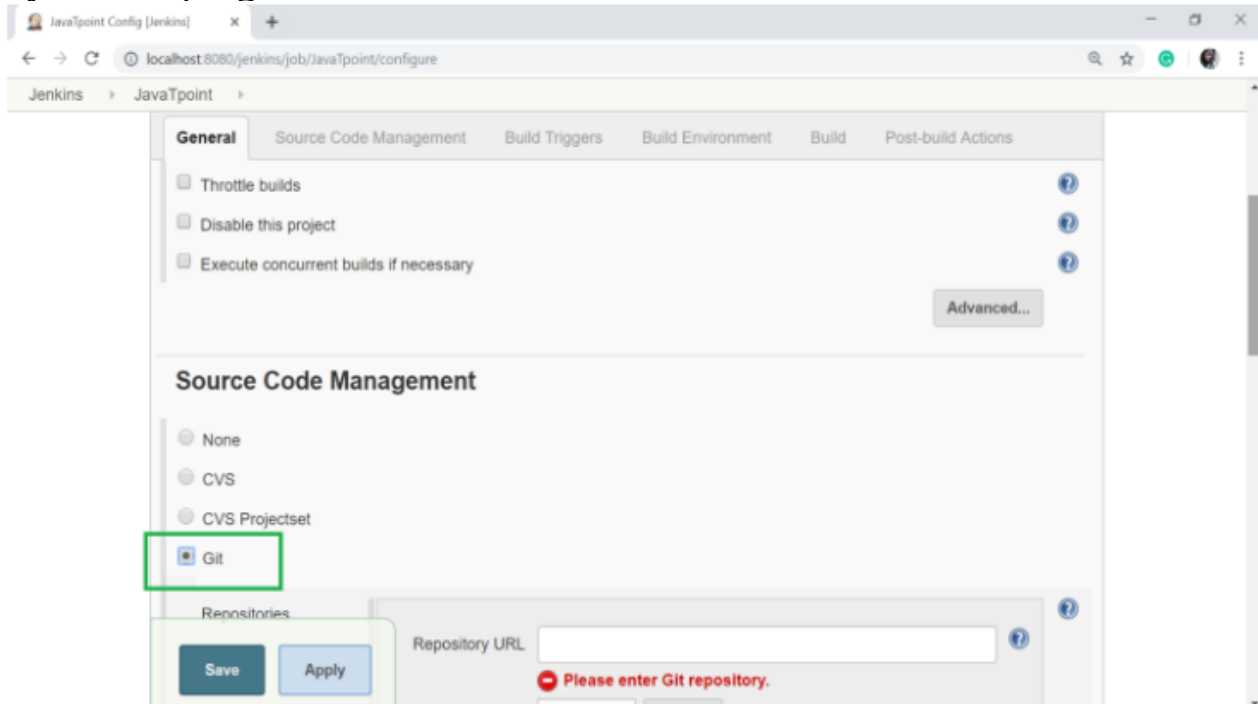


Step 2:

Integrating Jenkins Jobs with Github

* First create a new job in Jenkins, open the Jenkins Dashboard and click on "create new jobs".

- * Now enter the item name and select the job type. For example, item name is "javaTpoint" and job type is "Freestyle project".
- * Click on OK.
- * Once you click OK, the page will be redirected to its project configuration. Enter the project information:
- * Now, under the "Source Code Management" you will see the Git option, if your Git plugin has been installed in Jenkins:



Note: if the **Git** option does not appear, try to reinstall the plugins, followed by a restart into your Jenkins dashboard.

Step 3: Make sure you add the valid github repo.

Step 4: Setup the build step as per your requirement and build the job.

Make sure your jenkins server should have git installed in the machine before you start building any SCM job.

Lab 4: Github Automatic Trigger with Jenkins

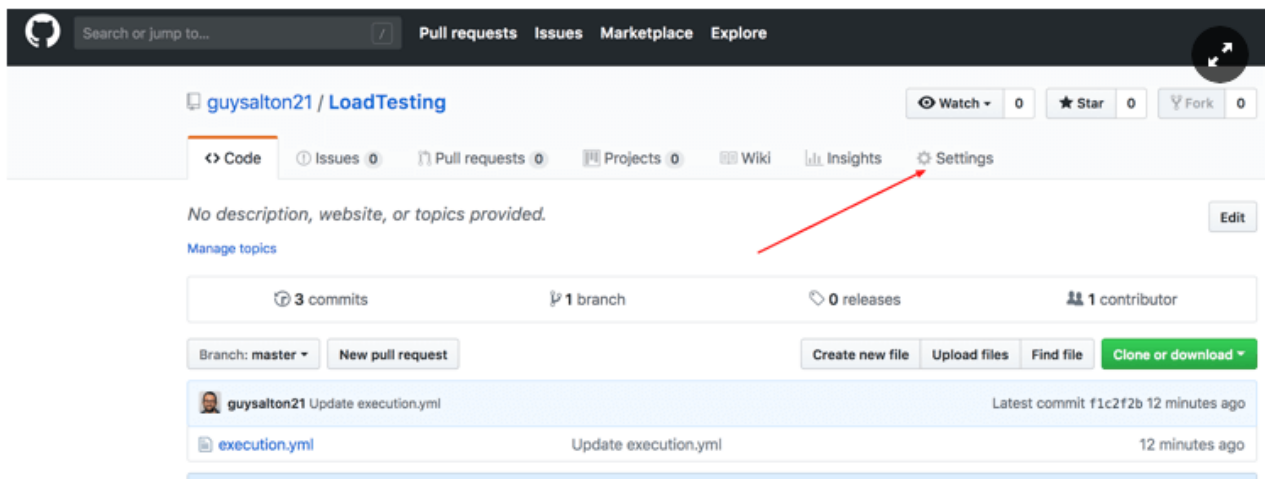
Ref: <https://www.blazemeter.com/blog/how-to-integrate-your-github-repository-to-your-jenkins-project>

The integration presented in this exercise will teach you to:

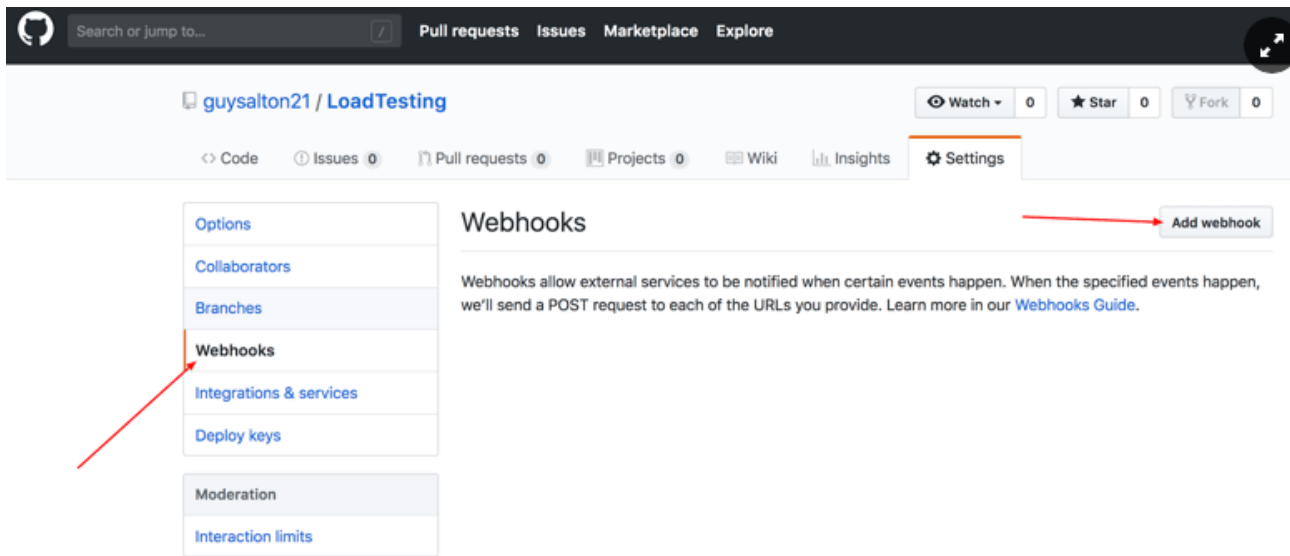
- 1) Schedule your build
- 2) Pull your code and data files from your GitHub repository to your Jenkins machine
- 3) Automatically trigger each build on the Jenkins server, after each Commit on your Git repository

Step 1: Configuring Github

Go to your GitHub repository and click on 'Settings'



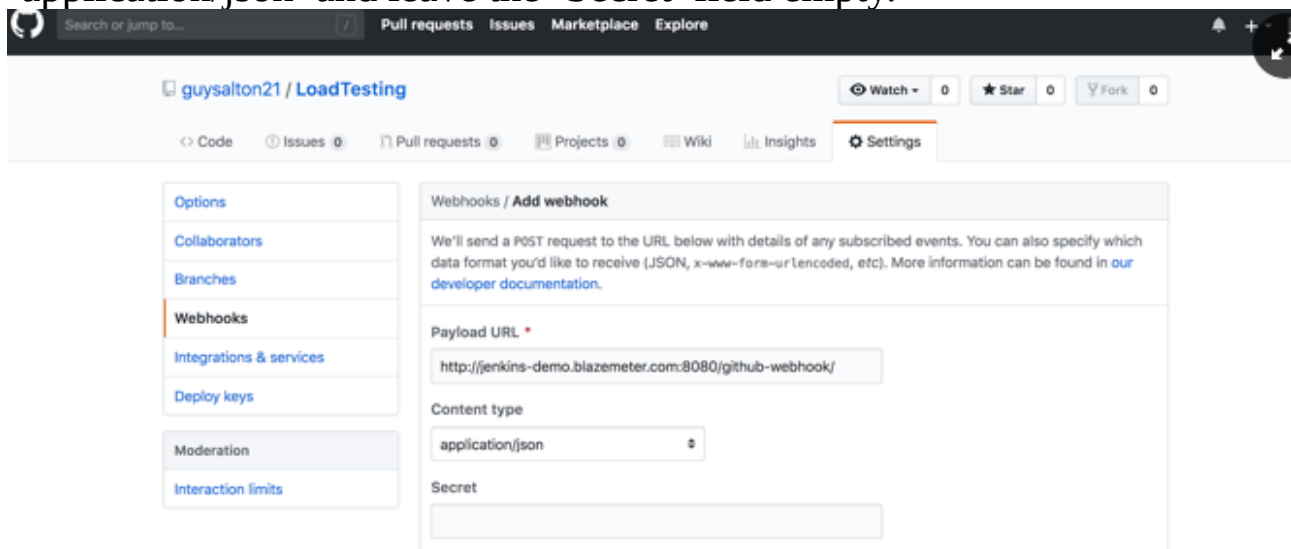
Step 2: Click on Webhooks and then click on ‘Add webhook’.



The screenshot shows the GitHub repository settings page for 'guysalton21 / LoadTesting'. The top navigation bar includes 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The repository name 'guysalton21 / LoadTesting' is displayed, along with 'Watch', 'Star', and 'Fork' buttons. The left sidebar contains a list of settings: 'Options', 'Collaborators', 'Branches', 'Webhooks', 'Integrations & services', 'Deploy keys', 'Moderation', and 'Interaction limits'. The 'Webhooks' option is highlighted with a red arrow. The main content area is titled 'Webhooks' and contains a description: 'Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).' A red arrow points to the 'Add webhook' button in the top right corner of the Webhooks section.

Step 3:

In the 'Payload URL' field, paste your Jenkins environment URL. At the end of this URL add /github-webhook/. In the 'Content type' select: 'application/json' and leave the 'Secret' field empty.

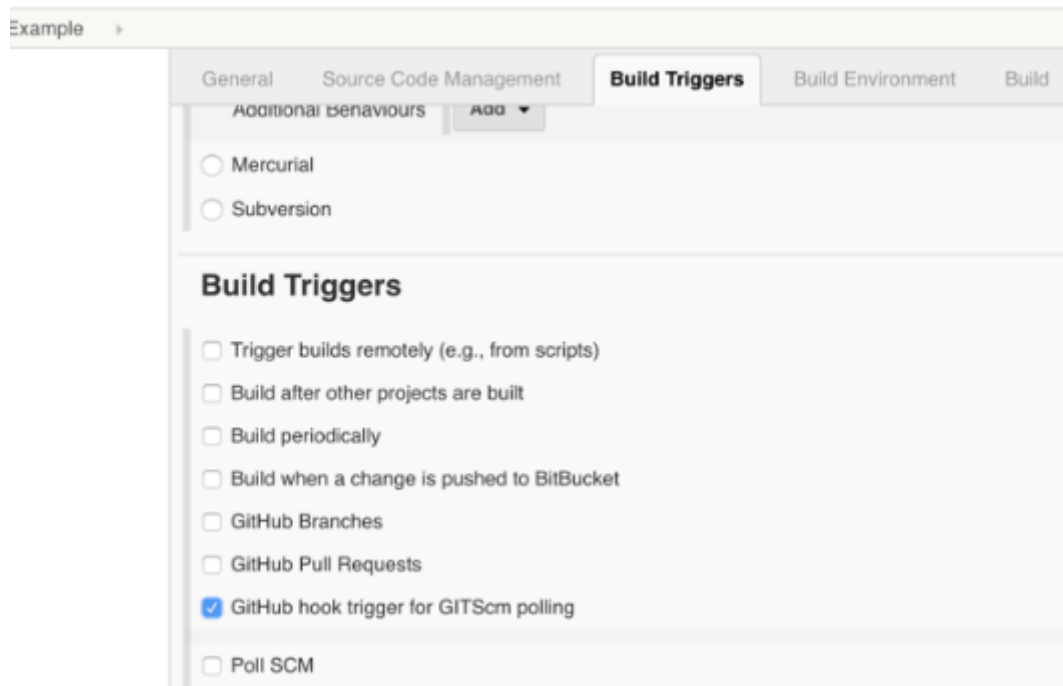


Step 4:

In the page 'Which events would you like to trigger this webhook?' choose 'Let me select individual events.' Then, check 'Pull Requests' and 'Pushes'. At the end of this option, make sure that the 'Active' option is checked and click on 'Add webhook'.

Step 5: Follow the steps of lab 3 on jenkins for adding git repo in a job with following changes

Add the trigger in the job to automatically trigger the job for commit in github.



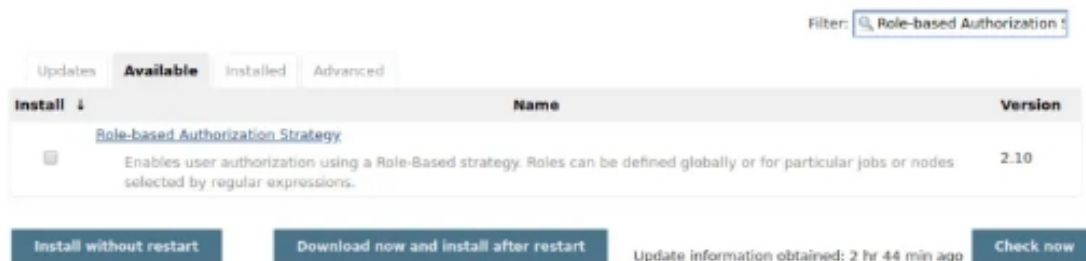
Add your build step as per requirement and do the changes in github repo to see the trigger automatically initiated.

Lab 5: Jenkins User and Roles

By default, when you create a user in Jenkins, it can access almost everything. In this exercise, we will cover how you can create fine-grained roles for proper access control to Jenkins Server. We will use Roles Strategy Plugin to achieve this.

Step 1:

Once Jenkins is live, login with the admin user account and navigate to Jenkins > Manage Jenkins > Manage Plugins > Available > Filter
Type “Role-based Authorization Strategy” in the filter box and hit enter.

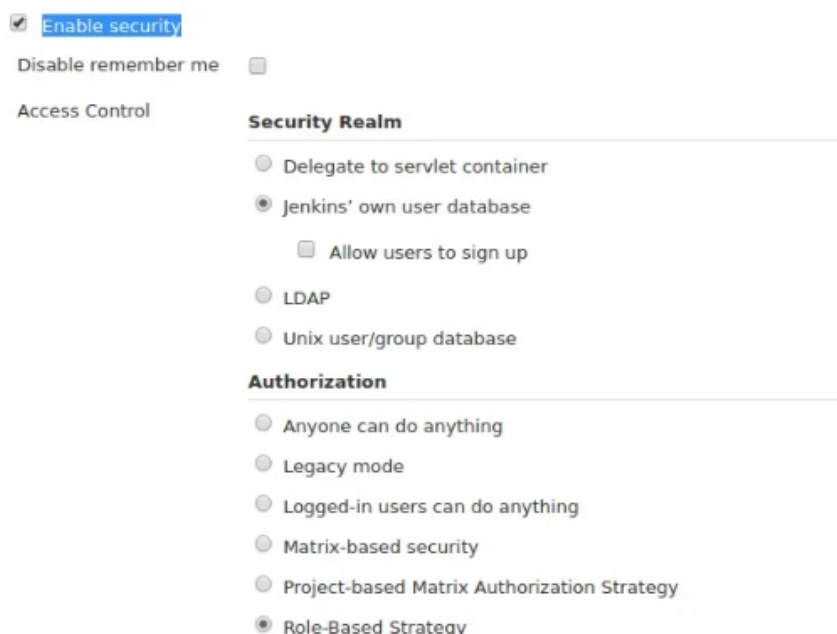


Select plugin and click the “Download now and install after restart“.

Step 2:

Enable Role-Based Strategy on Jenkins

After plugin installation, navigate to “Jenkins > Configure Global Security“. Tick Enable security and Role-Based Strategy then save settings.



Step 3 : Creating User Roles on Jenkins

Go to “Jenkins > Manage and Assign Roles > Manage Roles“.



Manage and Assign Roles



[Manage Roles](#)

Manage Roles



[Assign Roles](#)

Assign Roles



[Role Strategy Macros](#)

Provides info about macro usage and available macros

Step 4:

Provide role name to create on Role to add and click ‘Add“.

Tick appropriate values for your new role, in my case, we will be creating view only user so it will have.

Read under Overall

All under View

Job					Run			View				SCM	Lockable Resources	
Discover	Move	Read	ViewStatus	Workspace	Delete	Replay	Update	Configure	Create	Delete	Read	Tag	Reserve	Unlock
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

You can also create “Project roles” which will work for all projects with that matches specified pattern.

Step 5:

Assign roles to users

Go to “Jenkins > Manage and Assign Roles > Assign Roles“.

Global roles

User/group	admin	view
Joseph Mutai	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>

User/group to add

Add

Global roles

User/group	admin	view
Joseph Mutai	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>
user1	<input type="checkbox"/>	<input checked="" type="checkbox"/>

User/group to add

Add

Login as user with assigned role. Only projects granted should be visible.

Lab 6: Email notification with Jenkins

When you create jobs in Jenkins there must be some way to get the team and yourself notified about the build/test/deploy status. This is where Email Notifications can be used. Jenkins provide plugins to Send Emails. You just need to install and configure the plugin correctly.

This exercise will help you setting up the email notifications in few simple steps

You can configure email notifications in your jobs in two ways:

Default Email Notifier (Email-Notification)

Extended Email Notification (Editable E-mail Notification)

In Extended Email Notifications you can set triggers (e.g. build is unstable or before build), specify email subject, content and recipients.

Step 1

Installing Email Extension Plugin.

Open Jenkins using the following URL: <http://localhost:8080/> on any browser

Click on Manage Jenkins.

Click on Manage Plugins.

Select Email Extension and Email Extension Template Plugin and click Install Without Restart.

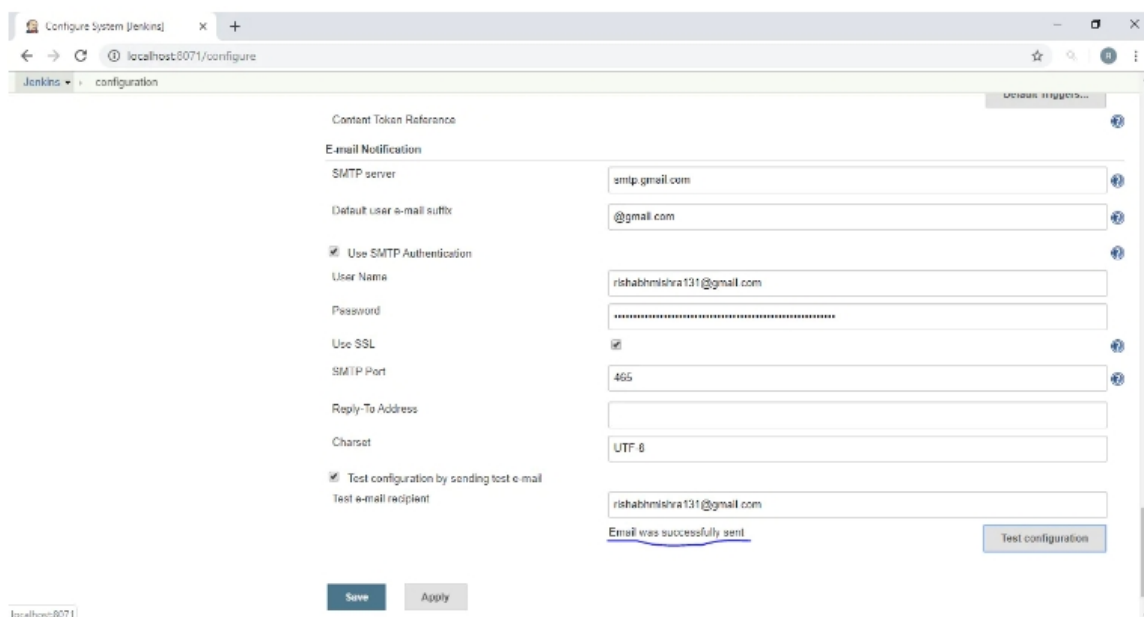
Once The Plugin Is Installed Let's Configure It With SMTP Servers So That Jenkins Emails Can Be Routed Via These SMTP Servers.

Step 2 - Configure Email Notifications

Click on Manage Jenkins and then Configure system

Scroll below till E-mail Notification and click on advanced. Setup up as shown in below screenshot and save it.

You can test configurations by entering recipient email address and clicking on test configuration. If all is good it will show a message - Email sent successfully. You may get error while testing configurations, below is possible errors and solution to it.



The screenshot shows the 'Configure System' page in Jenkins, specifically the 'Email Notification' section. The 'Advanced' tab is selected. The configuration includes:

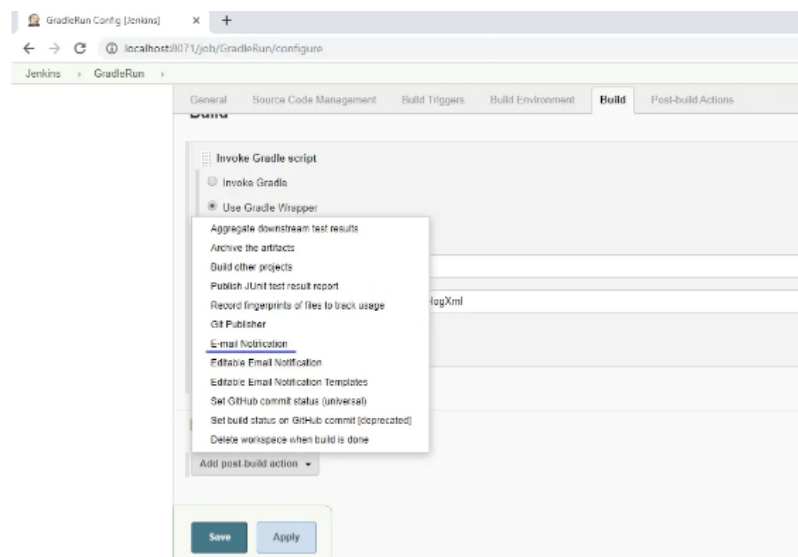
- SMTP server: smtp.gmail.com
- Default user e-mail suffix: @gmail.com
- ☒ Use SMTP Authentication
- User Name: rishabhmisra131@gmail.com
- Password: [Redacted]
- ☒ Use SSL
- SMTP Port: 465
- Reply-To Address: [Empty]
- Charset: UTF-8
- ☒ Test configuration by sending test e-mail
- Test e-mail recipient: rishabhmisra131@gmail.com

A message at the bottom right states 'Email was successfully sent' next to a 'Test configuration' button. At the bottom of the page are 'Save' and 'Apply' buttons.

Step 3:

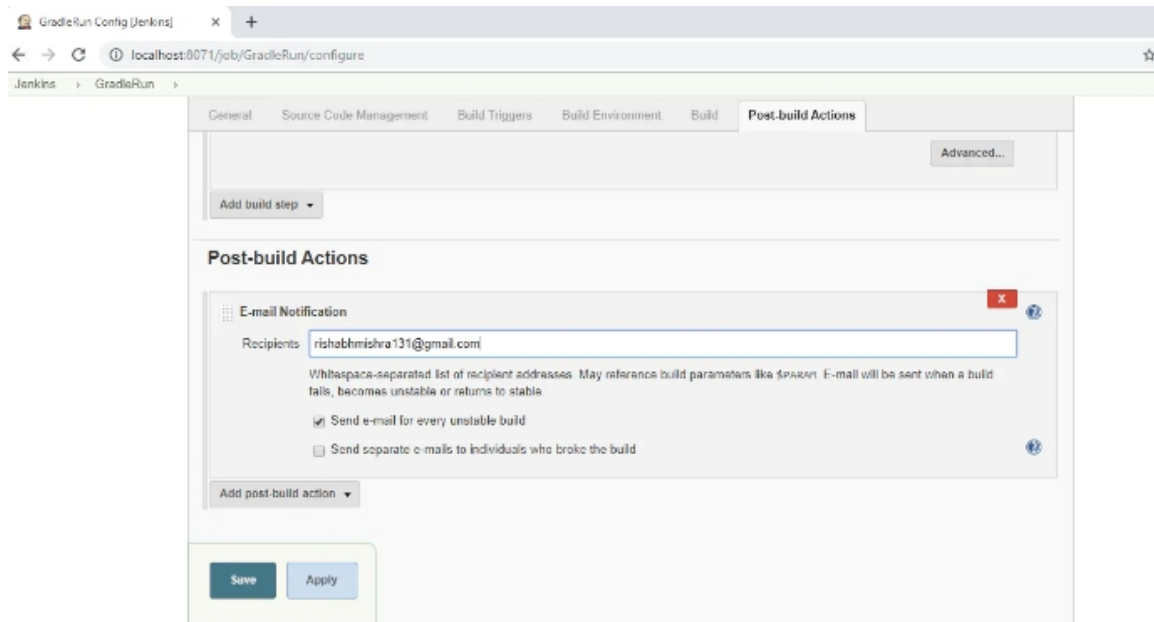
Configure the job for performing build operation

At the post build select the email notification



The screenshot shows the 'Configure' page for a Jenkins job named 'GradleRun'. The 'Post-build Actions' tab is selected. A list of post-build actions is shown, with 'Email Notification' highlighted. A dropdown menu is open, showing options for 'Email Notification', including 'Aggregate downstream test results', 'Archive the artifacts', 'Build other projects', 'Publish JUnit test result report', 'Record fingerprints of files to track usage', 'Git Publisher', 'Email Notification', 'Editable Email Notification', 'Editable Email Notification Templates', 'Set GitHub commit status (universal)', 'Set build status on GitHub commit (deprecated)', and 'Delete workspace when build is done'.

Step 4:
Enter recipients and check Send Email for every unstable build and save it.



The screenshot shows the Jenkins configuration page for a job named 'GradleRun'. The 'Post-build Actions' tab is selected. An 'E-mail Notification' action is added. The 'Recipients' field contains 'rishabhmrishra131@gmail.com'. The checkbox 'Send e-mail for every unstable build' is checked. The 'Save' button is highlighted.

GradleRun Config [Jenkins]

localhost:8071/job/GradleRun/configure

Jenkins > GradleRun >

General Source Code Management Build Triggers Build Environment Build **Post-build Actions**

Advanced...

Add build step

Post-build Actions

E-mail Notification

Recipients:

Whitespace-separated list of recipient addresses. May reference build parameters like \$PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.

☒ Send e-mail for every unstable build

☐ Send separate e-mails to individuals who broke the build

Add post-build action

Save Apply

Lab 7: Adding Jenkins Slave

A Jenkins master comes with the basic installation of Jenkins, and in this configuration, the master handles all the tasks for your build system.

If you are working on multiple projects you may run multiple jobs on each and every project. Some projects need to run on some particular nodes, and in this process, we need to configure slaves. Jenkins slaves connect to the Jenkins master using the Java Network Launch Protocol

Configure the Slave

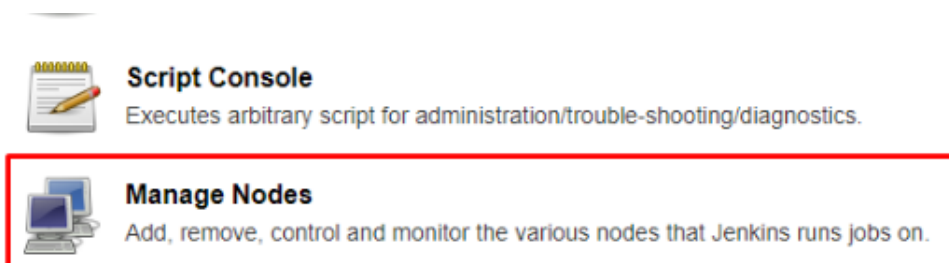
1) Login into your second ubuntu machine and run the following command to get it ready to work as slave

- * apt install openjdk-8-jdk
- * useradd -m ubuntu
- * passwd ubuntu
- * mkdir *home/ubuntu/jenkinsdata*

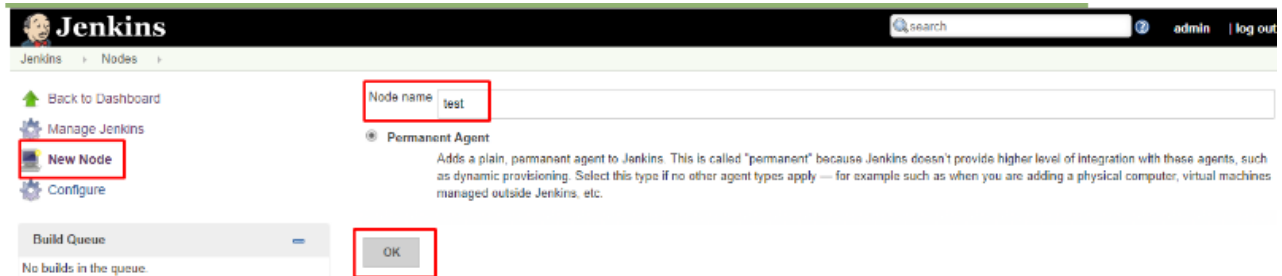
The user ubuntu will be used by jenkins master to connect to the slave and the directory created will be used in remote directory in master configuration.

Configure Master

- > Click on Manage Jenkins in the left corner on the Jenkins dashboard.
- > Click on Manage Nodes.



- > Select New Node and enter the name of the node in the Node Name field.
- > Select Permanent Agent and click the OK button. Initially, you will get only one option, "Permanent Agent." Once you have one or more slaves you will get the "Copy Existing Node" option.



Enter the required information.

Some required fields include:

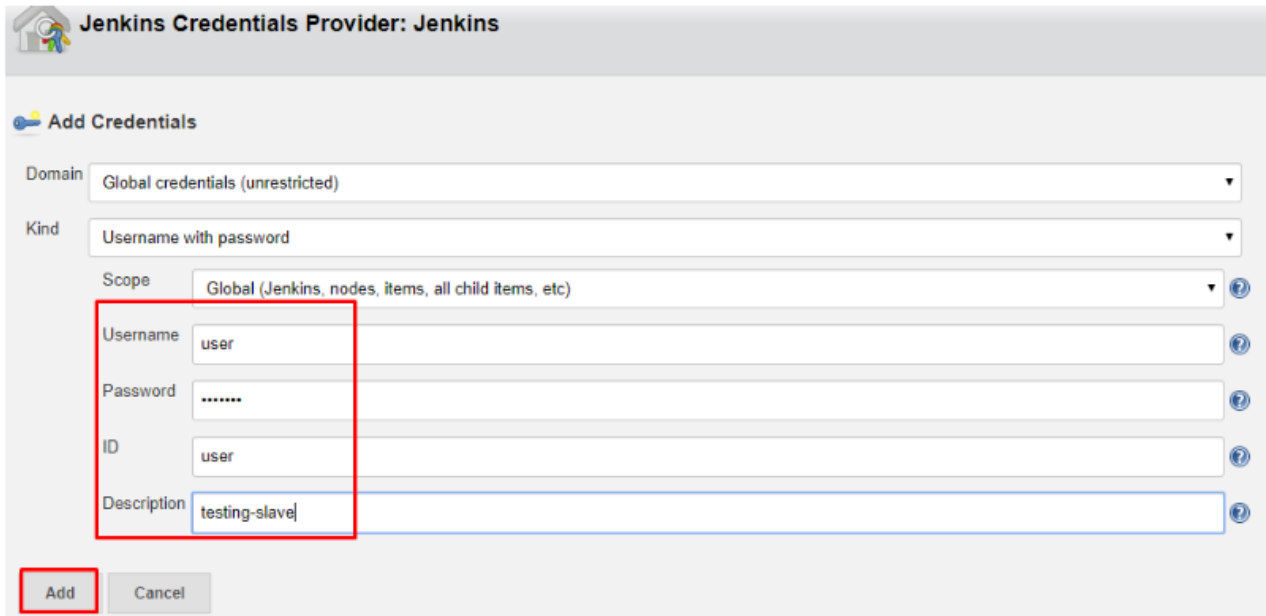
- * Name: Name of the Slave. e.g: Test
- * Description: Description for this slave (optional). e.g: testing slave
- * # of Executors: Maximum number of Parallel builds Jenkins master perform on this slave. e.g: #2
- * Remote root directory: A slave needs to have a directory dedicated to Jenkins. Specify the path to this directory on the agent. e.g: /home/
- * Usage: Controls how Jenkins schedules builds on this node. e.g: Only build jobs with label expressions matching this node.
- * Launch method: Controls how Jenkins starts this agent.

select: Launch agent agents via SSH

* enter the hostname or ip

* add ubuntu used credentials in jenkins provider

Select the dropdown menu to add credentials in the Credentials field



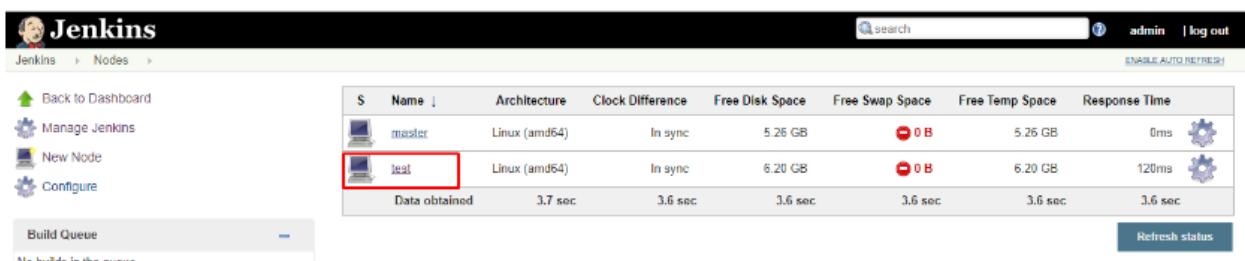
The image shows the 'Add Credentials' form in Jenkins. The form is titled 'Jenkins Credentials Provider: Jenkins'. It has a 'Domain' dropdown set to 'Global credentials (unrestricted)'. The 'Kind' dropdown is set to 'Username with password'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' field contains 'user', the 'Password' field contains '*****', the 'ID' field contains 'user', and the 'Description' field contains 'testing-slave'. The 'Add' button is highlighted with a red box.

.

Select the next dropdown to add the Host Key Verification Strategy under **Non verifying Verification Strategy**.

Select Keep this agent online as much as possible in the Availability field.

Click on save button to verify slave added on the master node.



The image shows the Jenkins Dashboard with the 'Nodes' view selected. The table lists the nodes 'master' and 'test'. The 'test' node is highlighted with a red box. The table shows the following data:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	5.26 GB	0 B	5.26 GB	0ms
	test	Linux (amd64)	In sync	6.20 GB	0 B	6.20 GB	120ms
Data obtained			3.7 sec	3.6 sec	3.6 sec	3.6 sec	3.6 sec

Lab 8: Jenkins Integration with Maven

Maven is build automation tool used basically for Java projects, though it can also be used to build and manage projects written in C#, Scala, Ruby, and other languages. Maven addresses two aspects of building software: 1st it describes how software is build and 2nd it describes its dependencies.

Step 1:

On the Jenkins master, install maven, maven is available as zip file and you can download the same from <https://maven.apache.org/download.cgi>

```
# mkdir "/var/lib/maven"  
# cd "/var/lib/maven"  
# wget  
https://apachemirror.wuchna.com/maven/maven-3/3.8.1/binaries/apache-maven-3.8.1-bin.tar.gz
```

untar the file

```
# tar zxvf apache-maven-3.8.1-bin.tar.gz
```

```
# cp -r apache-maven-3.8.1/* .
```

Step 2:

Setup JDK and maven information in jenkins for use.

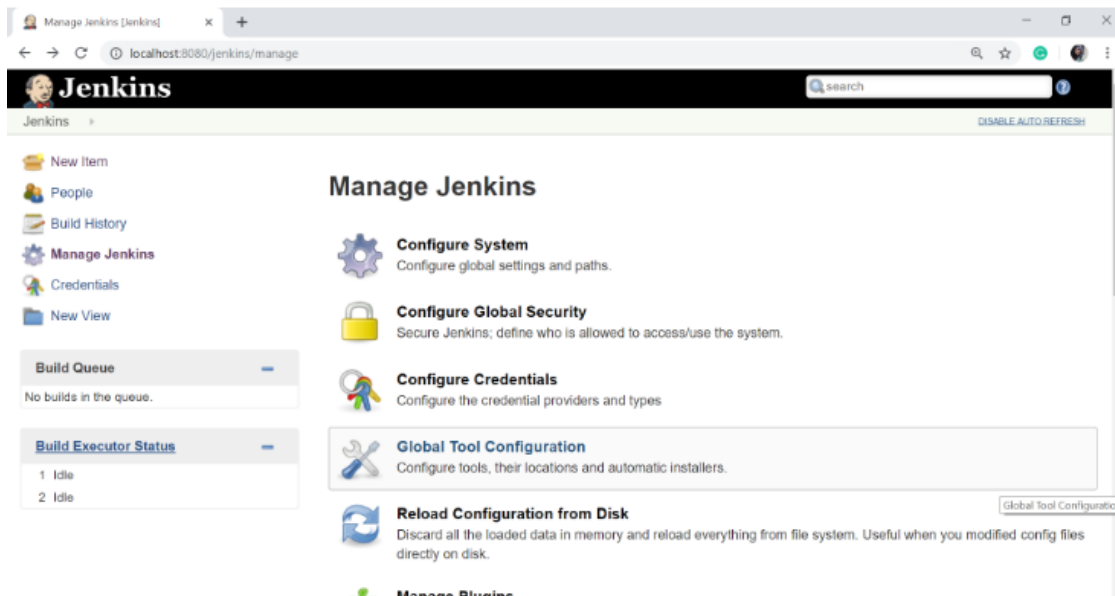
Verify the java and maven home in the system.

```
# java -version
```

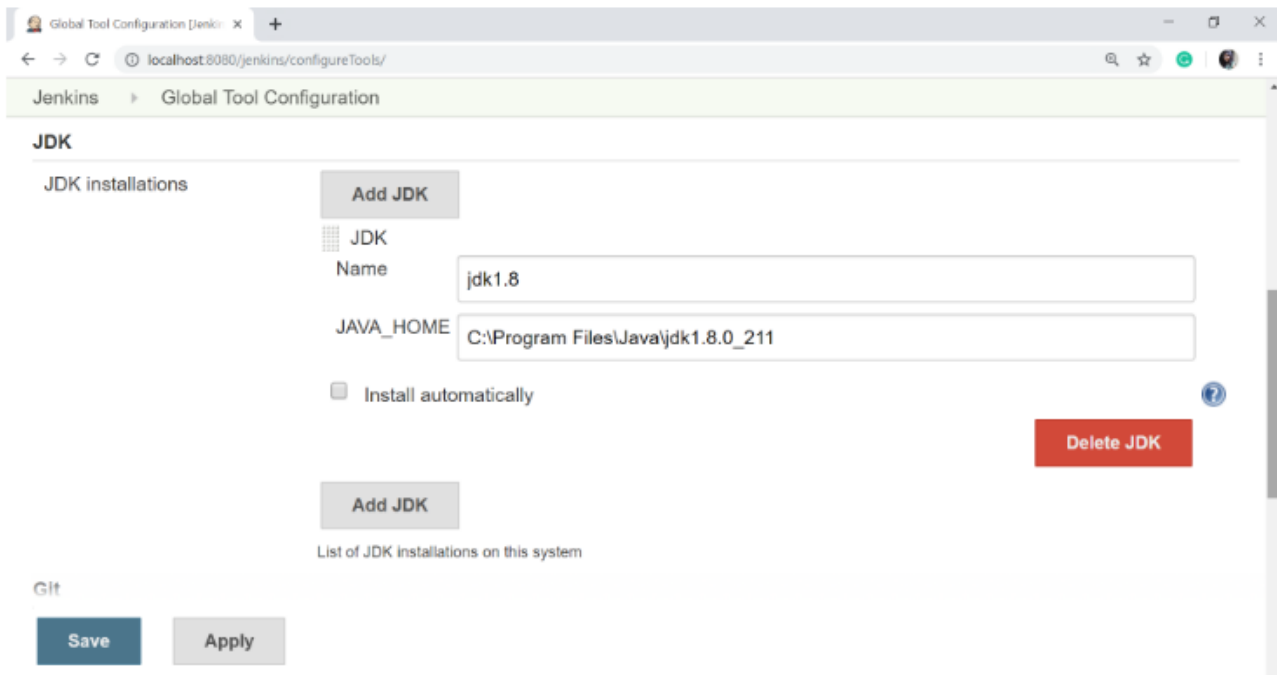
```
# /var /lib /maven / bin/ mvn -version
```

In the Jenkins dashboard (Home screen) click on manage Jenkins from the left-hand side menu.

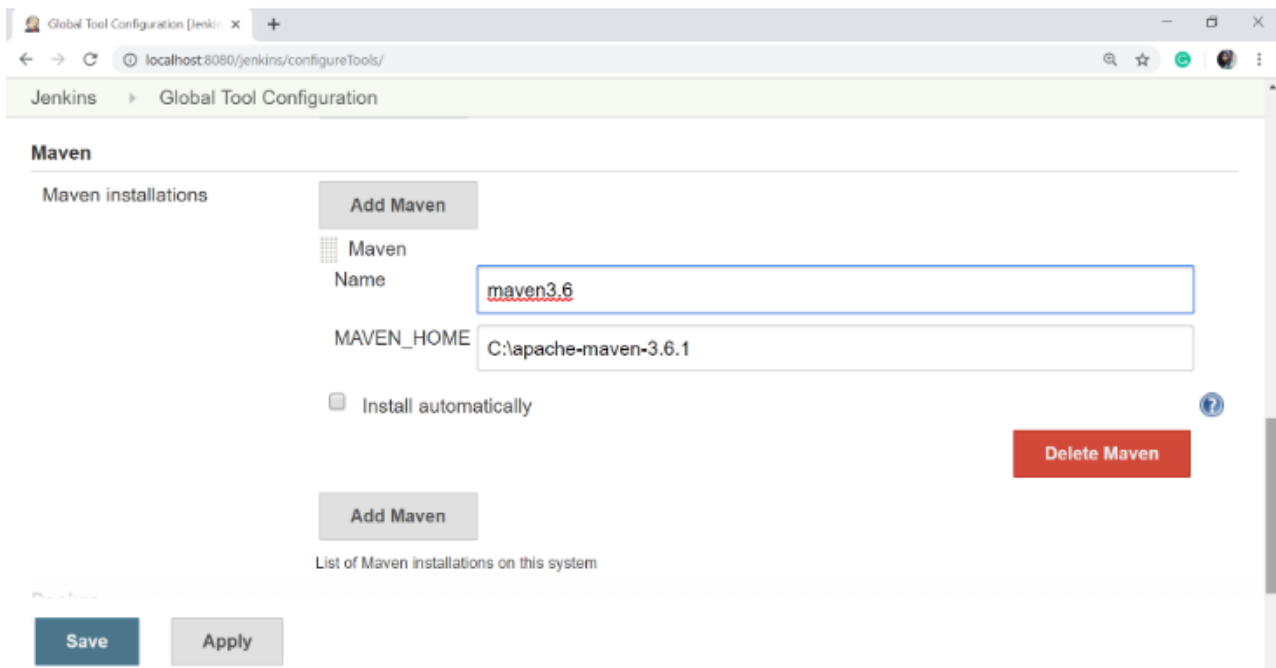
Click on "Global Tool Configuration" option.



To configure Java, click on "Add JDK" button in the JDK section. Give a Name and JAVA_HOME path, or check on install automatically checkbox.



And now, to configure Maven, click on "Add Maven" button in the Maven section, give any Name and MAVEN_HOME path or check to install automatically checkbox.



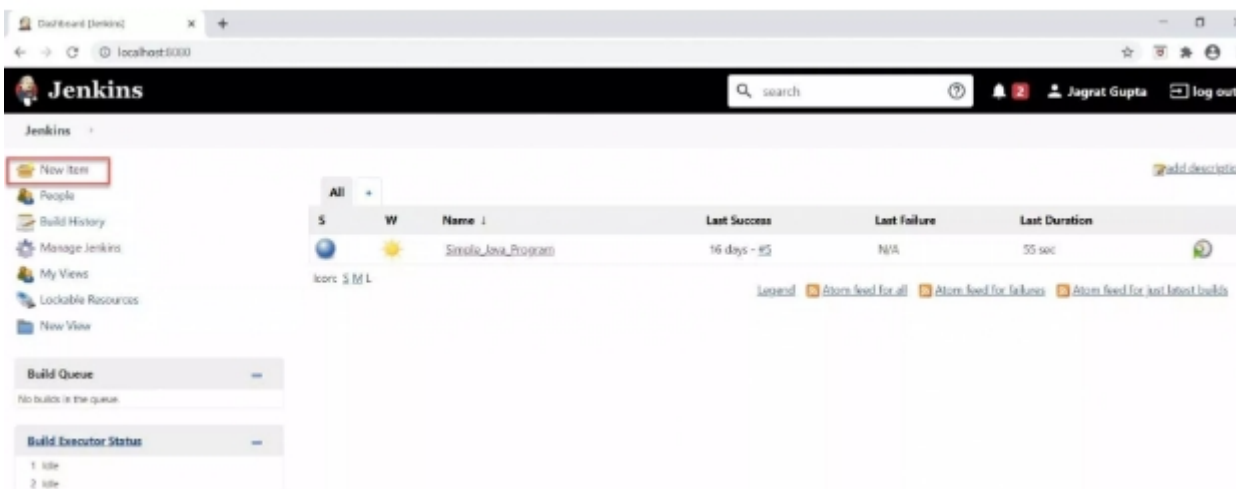
The screenshot shows the Jenkins 'Global Tool Configuration' page for Maven. The browser address bar indicates 'localhost:8080/jenkins/configureTools/'. The page title is 'Jenkins > Global Tool Configuration'. Under the 'Maven' section, there is a sub-section 'Maven installations'. It features an 'Add Maven' button, a table with one row for 'Maven', and input fields for 'Name' (containing 'maven3.6') and 'MAVEN_HOME' (containing 'C:\apache-maven-3.6.1'). There is an unchecked checkbox for 'Install automatically' and a red 'Delete Maven' button. At the bottom, there are 'Save' and 'Apply' buttons.

Name	MAVEN_HOME	Install automatically
maven3.6	C:\apache-maven-3.6.1	<input type="checkbox"/>

Then, click on the "Save" button at the end of the screen.

Now, you can create a job with the Maven project. To do that, click on the New Item option or create a new job option.

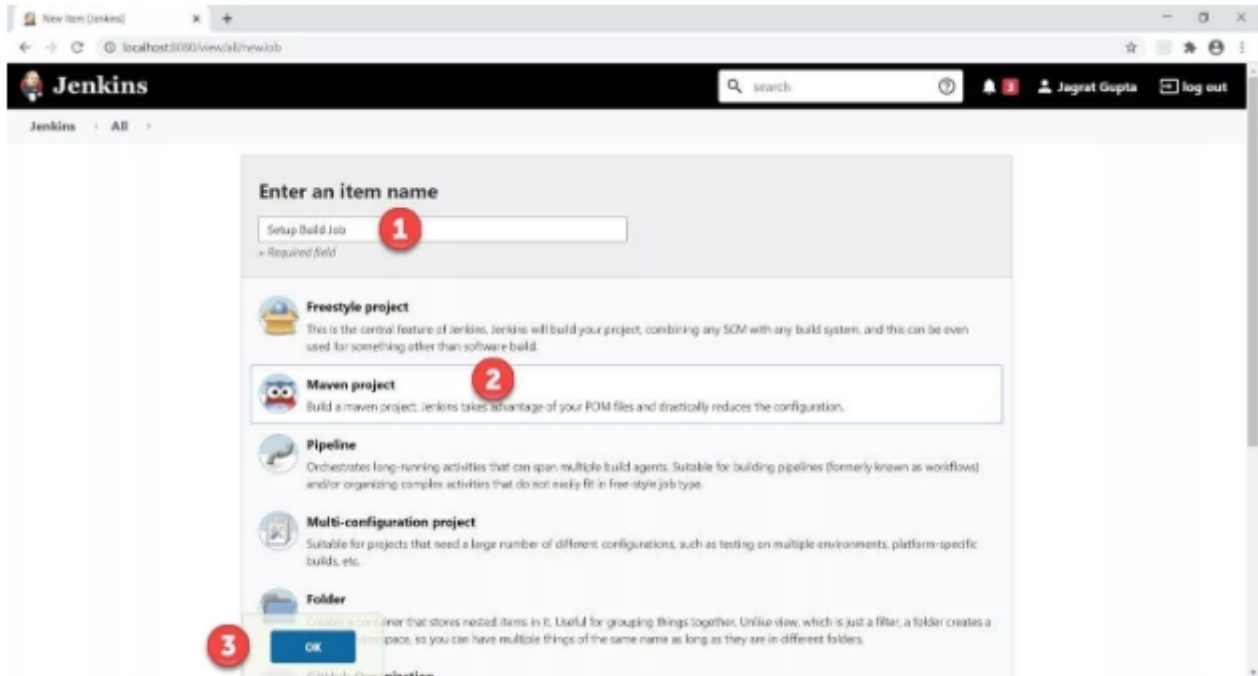
Step 3: Firstly, we need to create a job. To create this, click on the “*New Item*” option as highlighted below:



The screenshot shows the Jenkins Dashboard. The browser address bar indicates 'localhost:8080'. The page title is 'Jenkins'. The left sidebar contains a 'New Item' button, which is highlighted with a red box. Below it are links for 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. The main content area shows a table of jobs. The table has columns for 'S', 'W', 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. There is one job listed: 'Simple Java Program'.

S	W	Name	Last Success	Last Failure	Last Duration
		Simple Java Program	16 days - 15	N/A	55 sec

Step 4: Now, do the following steps to create a new maven project:
Give the Name of the project.
Click on the **Maven project**.



Step 4: Now, do the following steps to create a new maven project:
Give the Name of the project.
Select the scm option and put the github repo path which contains the maven code.

<https://github.com/pdurbin/maven-hello-world>

Step 5: Firstly, give the relative path of pom.xml in the Root POM textbox, as we do have the pom.xml at the my-app of the project, so we directly provided the file's name my-app/pom.xml.

Secondly, type “package” in the Goals to create the build jar file in the target directory of job workspace

Save the job and build it to check the build file in
\$JENKINS_HOME/workspace/\$JOB_NAME/target folder

Lab : Jenkins Maven and Docker Integration

Step 1: Prepare jenkins to run docker commands

> Login into jenkins machine and run the following command to install docker and configure jenkins user to run the docker command

```
# apt install docker.io
```

```
# usermod -aG docker jenkins
```

```
# service jenkins restart
```

```
root@ip-172-31-0-248:~#  
root@ip-172-31-0-248:~# apt install docker.io -y  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
docker.io is already the newest version (20.10.2-0ubuntu1~20.04.2).  
0 upgraded, 0 newly installed, 0 to remove and 58 not upgraded.  
root@ip-172-31-0-248:~# usermod -aG docker jenkins  
root@ip-172-31-0-248:~#  
root@ip-172-31-0-248:~#  
root@ip-172-31-0-248:~# service jenkins restart  
root@ip-172-31-0-248:~#
```

Step 2: Install the plugins in jenkins portal for integrating jenkins with Docker master

```
# login into portal
```

```
# click on manage jenkins
```

```
# click on manage plugins
```

```
# click on available and then search for “docker”
```

```
# install docker, docker-commons and cloudbees docker build & publish plugins.
```

Dashboard > Plugin Manager

Back to Dashboard
Manage Jenkins
Update Center

Search: docker

Updates Available Installed Advanced

Install	Name	Version	Released
<input type="checkbox"/>	Docker Cloud Providers Cluster Management and Distributed Build docker This plugin integrates Jenkins with Docker	1.2.2	4 mo 14 days ago
<input type="checkbox"/>	Docker Commons api-plugin docker Library plugins (for use by other plugins) Provides the common shared functionality for various Docker-related plugins.	1.17	11 mo ago
<input type="checkbox"/>	Docker Pipeline Deployment DevOps docker pipeline Build and use Docker containers from pipelines.	1.26	3 mo 14 days ago
<input type="checkbox"/>	Docker API api-plugin docker This plugin provides docker-java API for other plugins. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin	3.1.5.2	1 yr 2 mo ago

[Install without restart](#)
[Download now and install after restart](#)
 Update information obtained: 1 hr 38 min ago
 [Check now](#)

Step 3: Create a maven project with previous lab guidance.
Github repo should contains repo path for testing purpose.

<https://github.com/Java-Techie-jt/docker-jenkins-integration-sample>

Step 4: In build step, add the “docker build and publish step”

Add your dockerhub credentials for jenkins to read and publish the image to your account with new repository name.

Build

Docker Build and Publish X ?

Repository Name ?

⊘ Please set a name

Tag

Docker Host URI ?

Server credentials

- none - Add

Docker registry URL ?

Trigger the job and check the image being published in the dockerhub repo.